Appendix C

Recommendations to Vendors

Here we list assorted recommendations to designers and vendors of networking gear. Most of these messages are implicit in what we have said before, but it helps to spell them out.

We would, of course, be happier with a POSIX standard for many of these things.

C.1 Everyone

- *Keep it simple!* If it's complex, it's probably wrong.
- Build tools, not monolithic systems. Graphic user interfaces and menu-driven systems are not necessarily simpler, better, or easier to use, and they're generally poorer matches for tool output. How, for example, would you feed the output of a filter language compiler into an X11-based configuration system?
- Don't use conventional passwords.
- Ship it secure. Make people turn off security via explicit action. (And there's no need to make that action too easy.)

C.2 Hosts

- Add logging. Add lots of it. Create a uniform message format to describe all network connections and (especially) all access failures.
- Provide a standard network access control language that is used by all servers.
- Provide a standard subroutine library that applications can use to validate connections.
- Do user authentication in one place, and do it right. Let the system administrator decide what's right for his or her environment, but make sure that one-time passwords are supported.

- Some versions of *rlogind* and *rshd* have an option to ignore `.rhosts` files. We'd like all vendors to support that option, so that system administrators could decide which machines were trustworthy. But the `/etc/hosts.equiv` file format needs to be extended to permit triples of the form ⟨REMOTETHOST, *remoteuser*, *localuser*⟩.
- The mainframe world has long provided "user exits" in critical spots. This seems to be a lost art in the UNIX system world. Programs like *login* should be distributed as linkable object files, with documented calls to user exits at various spots. (But resist the temptation to provide the functionality via a shared library; shared libraries have been implicated in too many security holes, on too many different platforms.)
- Provide a mechanism to send unserviced TCP and UDP packets to a server. This mechanism could even replace *inetd*.
- Better facilities are needed to offer different services to different communities. For example, on a multihomed host, it would be nice to be able to bind a server to only one of the addresses, and to reject requests for that service if they arrive on a different interface.
- Enhance *telnetd* and *rlogind* so that they can run programs other than *login*. It would be even nicer if the protocols were implemented as STREAMS modules, which could just be pushed onto the network connection. That would make it a lot easier to install new network services.

C.3 Routers

- Filter mechanisms need to incorporate some sort of logging option.
- Most current filter languages are quite arcane. If a *good* language is too hard to parse in a router, build a compiler, and download the cryptic information. People do better with high-level languages.
- The distinction between incoming and outgoing TCP calls is critical; filter languages must support it.
- Filter packets on input instead of (or in addition to) output; otherwise, you lose valuable information.
- Routing protocols need filtering ability, too, on input and output.

C.4 Protocols

- Protocols that require the server to call the client back are very hard to manage in a firewall environment. To the extent possible, new protocols should involve only outbound calls from the client.

- Sequence numbers should be at least 32 bits, and chosen via a cryptographic random number generator.
- The distinction between outgoing calls and responses to them should be visible to packet filters, much as is possible with TCP.
- Protocols without fixed port numbers are problematic and should be avoided if possible.
- Integral mechanisms for indirection—`MX` records, the `$DISPLAY` variable for X11, etc.—are very useful hooks and should be used when possible. (Yes, we realize that this conflicts, to some extent, with the previous recommendation.)
- However, communicating the indirection via a part of the protocol, such as FTP's `PASV` subcommand, makes life harder.

C.5 Firewalls

- Provide copious logging.
- Support multiple styles of authentication, and let the administrators add their own.
- Provide an easy mechanism for administrators to add new services, even ones that require customer-written processes on the gateway machine.
- Support tunneling and encrypted tunneling.